공학 석사 학위 논문

Machine Learning-Based Automatic Generation of eFuse Configuration in NAND Flash Chip

머신 러닝 기반의 낸드 플래시 칩 eFuse 구성 생성 자동화 방법론

2019년 6월

서울대학교 대학원

컴퓨터 공학부

김 지 석

Abstract

Machine Learning-Based Automatic Generation of eFuse Configuration in NAND Flash Chip

Jisuk Kim
Department of Computer Science and Engineering
The Graduate School
Seoul National University

Post fabrication process is becoming more and more important as memory technology becomes complex, in the bid to satisfy target performance and yield across diverse business domains, such as servers, PCs, automotive, mobiles, and embedded devices, etc. Electronic fuse adjustment (eFuse optimization and trimming) is a traditional method used in the post fabrication processing of memory chips. Engineers adjust eFuse to compensate for wafer inter-chip variations or guarantee the operating characteristics, such as reliability, latency, power consumption, and I/O bandwidth. These require highly skilled expert engineers and yet take significant time. This paper proposes a novel machine learning-based method of automatic eFuse configuration to meet the target NAND flash operating characteristics. The proposed techniques can maximally reduce the expert

engineer's workload. The techniques consist of two steps: initial eFuse generation and eFuse optimization. In the first step, we apply the variational autoencoder (VAE) method to generate an initial eFuse configuration that will probably satisfy the target characteristics. In the second step, we apply the genetic algorithm (GA), which attempts to improve the initial eFuse configuration and finally achieve the target operating characteristics. We evaluate the proposed techniques with Samsung 64-Stacked vertical NAND (VNAND) in mass production. The automatic eFuse configuration takes only two days to complete the implementation.

Keywords : NAND Flash, eFuse, Optimization, Machine learning, Deep learning, VAE, Genetic Algorithm

Student Number : 2017-29459

Contents

Lists of Figures

Lists of Tables

I. INTRODUCTION

NAND flash chips can run faster and consume less power as NAND flash memory technology advances. However, owing to aggressive NAND cell scaling and fabrication, the worst-case cell characteristic has become critical affecting reliability, yield, and performance [1]. Memory engineers have developed test and chip configuration technologies to address poor cell characteristics. Through various test steps, e.g., measuring sampled chips during fabrication and real chip tests after fabrication, engineers determine the information for improving the wafer yield and chip performance, which we call chip information. It represents the operating condition and chip status, e.g. the analog bias level, defect location found during the test steps, inter-chip variation information on a wafer, etc., which are applied to adjust the operating characteristics of NAND flash chips and finally determines the reliability, latency, and power consumption of NAND flash chips. The chip information is stored in specific NAND cells that are not accessible to the user. During system boot-up, the NAND flash chip reads the chip information from the cells in its power-up sequence and stores it in a latch-structured register circuit called eFuse (electric fuse) thereby enabling the associated circuits to access chip information [2].

Table I shows the chip information. A typical NAND flash chip requires thousands of eFuses, and 80 to 90 percent of the eFuses are easily determined by the automatic trim method. To ensure the normal circuit

Table I. NAND Flash Chip information

Item	eFuses Ratio	Optimization
Bad Block & Repair Column Analog Bias Level Trim Clock Generator (Oscillator) Trim Chip Variation Compensation (Calibration) I/O Speed Tuning	**80 ~ 90 %**	**Automatic Trimming**
Optimized eFuse configuration for Reliability Optimized eFuse configuration for Power Reduction Optimized eFuse configuration for Latency	**10 ~ 20 %**	**Manual Heuristic**

operation of an analog circuit (e.g., voltage generator), the test equipment sweeps the value of the corresponding eFuse and selects the best eFuse value which ensures the reference voltage determined at the design stage. Another example is the test equipment that localizes defective cells and stores the location information in special cells, which are transferred, during a power-up sequence, to the eFuse that manages repair circuits.

The design time for eFuse configuration is dominated by configuring the remaining eFuses (10~20% of total eFuses), which we call *hard eFuses*, for the NAND operating characteristics, e.g., reliability, latency, power consumption. The operating characteristics are determined by the inter-related behavior of dozens of analog and digital circuits configured by the associated hard eFuses. In real test stages, such hard eFuses are configured through the manual optimization process, which requires numerous trials of real chip measurements and eFuse value adjustments, hence the significant duration of design time. The lengthy design time problem becomes

extremely critical as NAND flash chips are embedded in various products such as servers, PCs, automotive, mobiles, and embedded systems. Each product requires a specific eFuse configuration, which is time-consuming. Moreover, when the flash chip fabrication technology is changed (improved or modified), a new eFuse configuration is required. Thus, it is critical to reduce the design time of hard eFuse configuration and guarantee the quality of operating characteristics in all the products and process technologies. In our work, we aim at reducing the design time of hard eFuse configuration by adopting machine learning approaches.

In this paper, we propose a new method of automating the eFuse configuration of NAND flash chips. It consists of two stages. In the first step of the initial eFuse generation, we apply the variational autoencoder (VAE) [3] method to generate an eFuse configuration that is likely to meet the target characteristics. In the second step of the eFuse optimization, the genetic algorithm (GA) [4] method improves the initial eFuse configuration thereby finally meeting the target operating characteristics.

The remainder of this paper is as follows. Section II introduces the architecture of NAND flash chip and eFuse usage. Section III explains the basic idea. Sections IV and V describe the two-step eFuse generation methods. Section VI reports experimental results based on the mass production of NAND flash chip. Section VII reviews related work. Section VIII concludes the paper.

II. BACKGROUND

2.1. NAND Flash Block Architecture

A NAND flash memory chip consists of blocks. Figure 1 shows the internal architecture of the block. A block consists of tens of word lines (WLs) each of which corresponds to a page [5]. Typically, a page in 3D vertical NAND (3D VNAND) contains 64k or 128k cells (=8KB or 16KB). When accessing (reading or writing) a page, we apply a voltage level to WL and each of all the cells on the selected page is accessed.

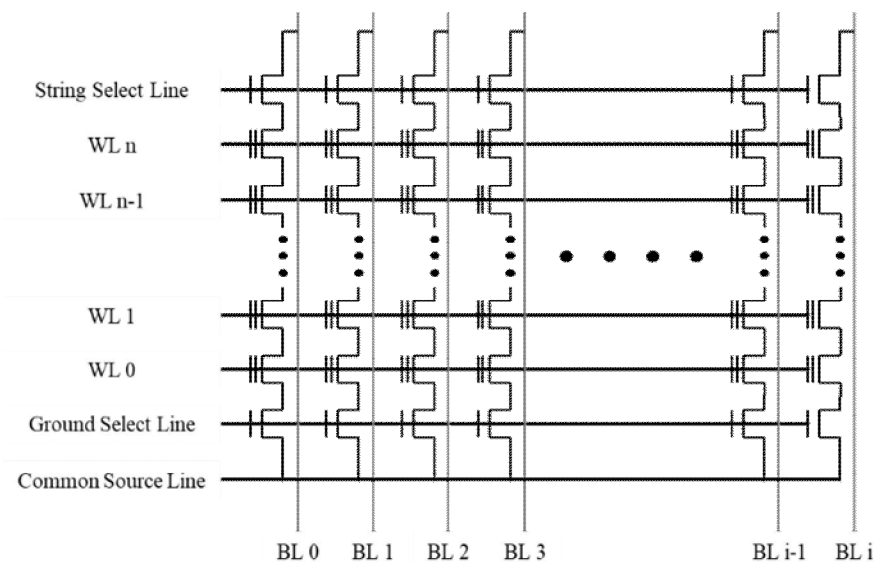


Figure 1. NAND flash block architecture

2.2. NAND Cell V_{th} Distribution

Figure 2 illustrates the distribution of threshold voltage, V_{th} in a page of the triple-level cell (TLC) NAND flash memory. The TLC NAND flash memory, stores 3 bits in a single NAND cell (000 to 111). Depending on the stored data, the NAND cells are grouped into eight states by the threshold voltage. In the case of the TLC cells, in order to read the entire contents of a page, at least seven different voltage levels are individually applied to the WL. Each voltage level called *verify level* distinguishes the cells into two groups (cells having lower/higher threshold voltage than the current WL voltage level). In Figure 2, for example, in order to read 1st data from a page, a verify level between E0 and P1 is applied to WL, grouping the E0 state into '1' group and the rest of the states into '0' group. We call such an operation a *verify operation*. Note that each BL is equipped with latches to store the group information of each verify operation finally to obtain the stored data. In the P1 to P7 states grouped

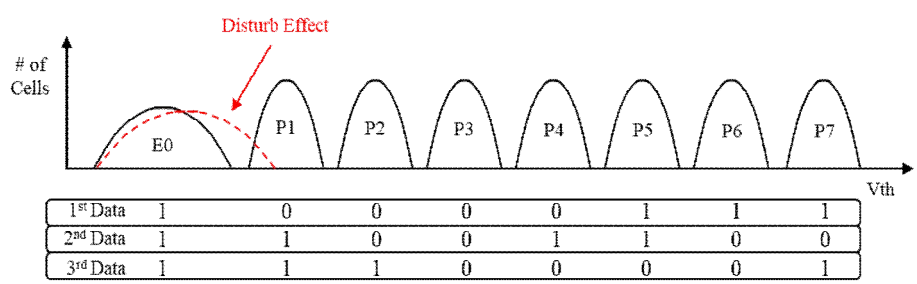
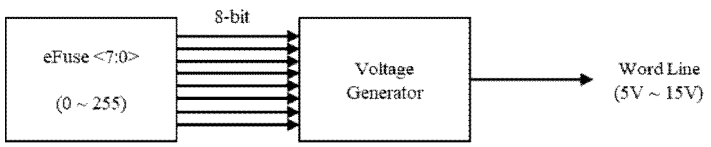


Figure 2. Cell V_{th} distribution of TLC NAND flash

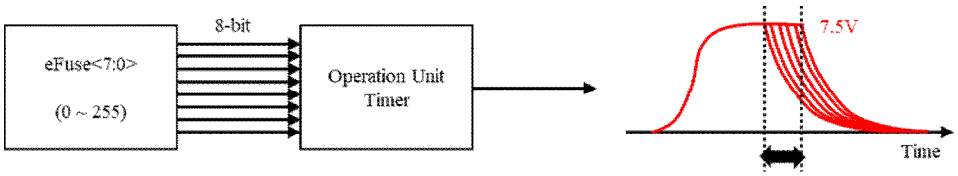
as '0' data in the first verify operation, another verify level between P4 and P5 is applied, and P5 to P7 states are grouped into '1' group, while the other states are grouped to '0' group. Generally, the smaller the width of the programmed cell state (P1 to P7), the lower the V_{th} of the upper bound of the erase (E0) state, and thus the NAND cell has better characteristics. Figure 2 exemplifies a wide V_{th} distribution (in a dashed line) of E0 state due to disturb. The wider V_{th} distribution a cell state has, the higher the probability of error is obtained in the read operation [6].

2.3. eFuse Operation of NAND Flash Chip

Figure 3 shows two examples of eFuse usage. An eFuse has a size of 5 to 10 bits, typically 8 bits. Each bit's information is stored in a latch



(a) eFuse usage for analog voltage generator



(b) eFuse usage for operation unit timer

Figure 3. Example of eFuse in NAND flash chip

circuit. Each eFuse is placed near the circuit that requires the information, enabling quick access during the chip operation.

Figure 3(a) shows an 8-bit eFuse used by an analog voltage generator. The 8-bit eFuse can contain a value within the range of 0 to 255 and the analog voltage generator circuit connected to the eFuse, outputs an analog level matching the eFuse value. Assuming the eFuse value is incremented sequentially from 0 to 255, the analog level output can be linearly increased, e.g., from 5V to 15V. Figure 3(b) shows another example of an eFuse used to adjust the operation's unit time. In this case, an 8-bit eFuse was used to adjust the operation time (e.g., from 40 ns to 10 μs at a unit of 40 ns).

III. BASIC IDEA AND BACKGROUND

3.1. Basic Idea

Automating the configuration of hard eFuses is challenging because multiple eFuses affect the operating characteristics, e.g., V_{th} cell distribution, and their relationship is very complicated. In addition, the design space is extensive considering that we need to handle hundreds of hard eFuses affecting various operating characteristics. That is why the configuration of hard eFuses has been manually performed by expert engineers.

In our work, we engage in solving the automation problem through a machine learning approach. First, our problem is defined as follows: Given the target operating characteristics, we are to generate the hard eFuse configuration that satisfies the target characteristics. The operating characteristics include M (= 24 in our experiments) items such as the width of NAND V_{th} cell distribution (PSUM), upper bound of the erase V_{th} cell distribution (Disturb), and program time (tPROG). To train the machine learning-based solution, we prepare the training dataset as follows: We obtain K (= 50,000) training data, each of which consists of N (= 32) randomly configured 8-bit eFuses and the measurement data of their associated M operating characteristics. For each training data, we configure a real NAND flash memory chip with the randomly generated eFuses and measure the operating characteristics of the chip.

In a naive machine learning approach, we can attempt to train a neural network [7], e.g., multi-layer perceptron (MLP) with the training dataset

consisting of pairs of eFuse configuration (output) and the associated operating characteristics (input). According to our experiments, such a naive approach has a critical problem that requires a large amount of training data due to overfitting [8]. Considering that the training data are obtained by configuring and measuring real NAND flash chips, the naive approach is prohibitively expensive, i.e., requires lengthy design time.

In order to reduce the amount of required training data, or for the same amount of training data to offer better configurations than the naive approach, we judiciously exploit machine learning technology, specifically, variational autoencoder (VAE) [3]. A VAE allows the reduction of training data by transforming its input, i.e., the operating characteristics, to a smaller amount of data, i.e., compressed data while managing the key relationships among the operating characteristics. Then, we train a small neural network, which accepts the compressed data as an input, and produces as an output the eFuse configuration that we call *initial eFuse configuration*. Considering that the amount of required training data is proportional to the size of the neural network, the VAE has the potential of reducing it for the network of eFuse generation. In addition, our experiments prove that our VAE-based solution offers better eFuse configurations than the naive approach due to the inherent characteristics of the VAE solution such as regularization effects [7] as will be explained in the next section.

After we obtain the initial eFuse configuration with our VAE-based solution, we optimize this configuration to meet the target operating characteristics. To do that, we apply the genetic algorithm (GA). The initial configuration presents an acceptable initial solution to the GA and thereby

allows us to manage the search space small and make the GA computationally efficient. Finally, our two-step approach can automatically produce the configuration of hard eFuses, satisfying the target operating characteristics.

3.2. Background: Variational Autoencoder

Figure 4 shows the VAE structure and operating principle. It consists of encoder and decoder neural networks. For a given input (in the original applications, an image, and in our work, the desired operating

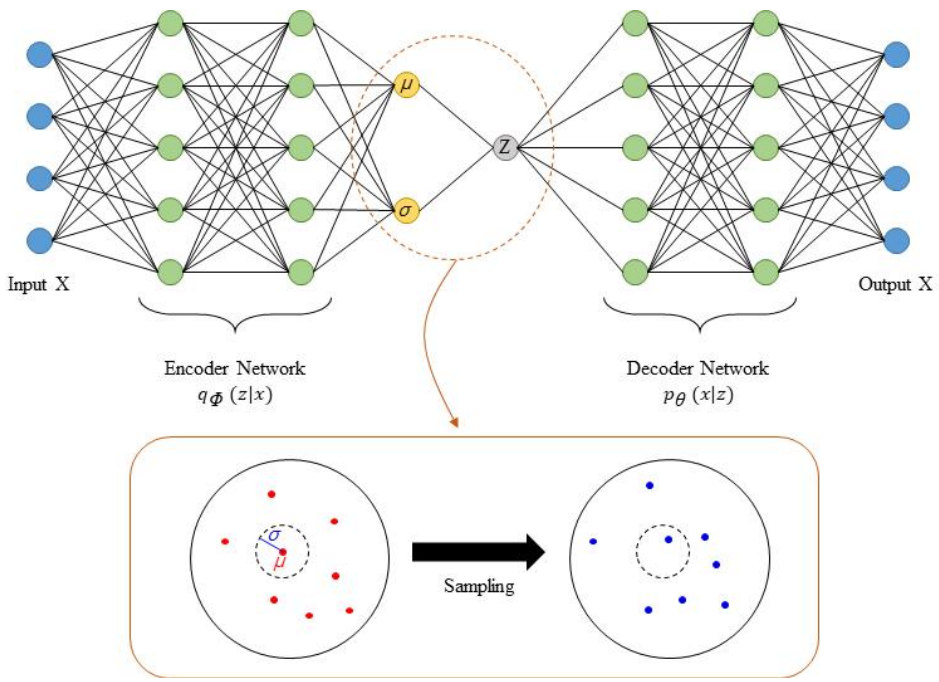


Figure 4. Structure of variational autoencoder (VAE)

characteristics), the encoder transforms the input to a low-dimensional latent vector Z. The VAE is trained to make the latent vector Z have a Gaussian distribution. For each input, as Figure 4 shows, the encoder network first obtains the mean and variance vectors of the Gaussian distribution, then samples a Z vector from the distribution. Thus, the output of the encoder is a sample of this distribution determined by the given input. This sample is used as an input to the decoder network. The decoder attempts to restore the original input data (in our case, the desired operating characteristics), which is called reconstruction. The VAE is called a generative model because new data (e.g., new images in the original VAE) can be generated by changing the latent vector Z.

The VAE is trained to maximize the probability of producing the same output as the given input. Given input x, the probability is represented as follows:

$$p(x) = \int p(z)\ p(x|z)\ dz \quad (1)$$

In (1), all Z must be considered for integration, which cannot be computed if the distribution of Z is unknown and the dimension of Z is large. The VAE uses variational inference (VI) to address this problem [9]. VI is a method of solving the problem by approximating the original distribution with a simple distribution $q(z)$ (e.g. Gaussian distribution). The process of maximizing $p(x)$ using VI is as follows:

1. To generate the output X, we first obtain the sample Z that follows the simple probability distribution $q(z|x)$. It is important to find a distribution of $q(z|x)$ similar to the distribution of $p(z|x)$. For example in (1), $p(z|x)$ is not directly computable because $p(z|x) = p(x|z)p(z)/p(x)$.

2. Equation (2) shows the log probability, $\log p(x)$ where θ and ϕ represent the model parameters of decoder and encoder networks, respectively. The equation shows that the log probability, rearranged using Bayes rule, consists of two terms, $\log p(x|z)$ and two Kullback-Leibler (KL) divergence [10]. The KL divergence represents the distance between two probability distributions and is always non-negative. Calculating (2) is also intractable due to $p(z|x)$ in the second KL divergence (in red). However, it is a negligible term because the two probability distributions of the term approach to each other, i.e., their KL divergence approaches zero through training. Thus, in order to maximize (2), we have to maximize the first two terms called the *evidence lower bound* (ELBO) shown in blue.

$$
\begin{aligned}
\log p(x) &= \mathbf{E}_z\left[\log \frac{p(x|z)p(z)}{p(z|x)}\right] \\
&= \mathbf{E}_z\left[\log \frac{p(x|z)p(z)}{p(z|x)}\frac{q(z|x)}{q(z|x)}\right] \\
&= \mathbf{E}_z[\log p(x|z)] - \mathbf{E}_z\left[\log \frac{q(z|x)}{p(z)}\right] + \mathbf{E}_z\left[\log \frac{q(z|x)}{p(z|x)}\right] \\
&= \mathbf{E}_z\left[\log p_\theta(x|z)\right] - D_{KL}[q_\phi(z|x)||p(z) \\
&\qquad + D_{KL}[q_\phi(z|x)||p(z|x)] \qquad (2)
\end{aligned}
$$

3. Equation (3) shows the ELBO which is the target function to maximize in the VAE's training. The left and right terms can be approximated using the VAE's decoder (left) and encoder (right) networks. ELBO is maximized by back propagation in encoder and decoder neural networks [7].

$$\mathcal{L}(x|z) = \log p_{\theta}(x|z) - D_{KL}[q_{\phi}(z|x)||p(z)] \quad (3)$$

For more details of the VAE operation such as re-parameterization techniques, refer to [3].

IV. INITIAL eFuse GENERATION: VAE-BASED DUAL NETWORK

Figure 5 shows our proposed network called the VAE-based dual network which accepts as input the target operating characteristics and as output generates the corresponding initial eFuse configuration. It consists of two neural networks, the VAE and MLP. The first network follows the VAE architecture, i.e., consists of encoder and decoder sub-networks. The encoder accepts as input the operating characteristics and generates (i.e., compresses the input) a latent vector Z with a smaller size than the input operating characteristics. The decoder accepts as input, the latent vector Z, and tries to produce as output the same operating characteristics as the input. The second network follows the MLP structure and produces the

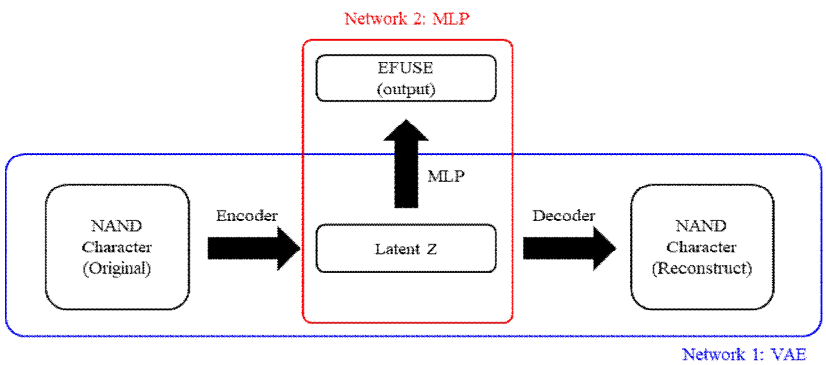


Figure 5. Structure of VAE-based dual network

initial eFuse configuration from the latent vector Z associated with the input operating characteristics. Thus, to generate an initial eFuse configuration, the target operating characteristics is provided as input to the encoder, which produces the latent vector Z. Then, the MLP accepts the latent vector as an input to produce the initial eFuse configuration.

The VAE-based dual network is trained in two steps. First, the VAE network is trained in an unsupervised manner. Given the training dataset, we utilize only the operating characteristics. The VAE network accepts as input a set of operating characteristics generates a latent vector Z and tries to produce, as output, the same operating characteristics. Thus, the training loss is defined to be the difference between the input and output. The training adopts the traditional back-propagation to update the weight parameters of the encoder and decoder sub-networks. Note that the data size of latent vector Z (= 10 in our experiments) is much smaller than that of the input operating characteristics (= 24). Thus, the encoder compresses the input data (into the latent vector Z) while minimizing information loss (the difference between the input and output of the VAE).

Once the VAE network training is completed, the second network, the MLP, is trained. First, we update the training dataset by replacing the operating characteristics with the associated latent vector obtained by the encoder of the VAE network. Then, we train the MLP with the updated training dataset. Thus, the MLP accepts as input the latent vector and produces an eFuse configuration. The training loss of the MLP is defined as the difference, or the cross-entropy [11] between the generated eFuse configuration (e.g., a 32-dimensional vector of 8-bit values, each representing

an 8-bit eFuse value in our experiments) and the label, i.e., the eFuse configuration in the training dataset. After obtaining the training loss, back-propagation is performed to update the weight parameters of the MLP network, for each training data.

We mentioned previously that the VAE-based dual network can benefit from the regularization effects of the VAE network to prevent overfitting and require a reduced amount of training data. As explained in Section III, the VAE network samples from the Gaussian distribution. We performed multiple epochs[1] of training. Thus, the same training data were learned multiple times. In such a multi-epoch training, the sampling can produce different latent vectors for the same input training data across epochs thereby providing the effects of regularization[2] [7].

---

1) During one epoch, each of all the training data is learned once.

2) Sampling the VAE has the same characteristics as noise injection, which is a kind of regularization techniques.

V. eFuse OPTIMIZATION: GENETIC ALGORITHM

In order to meet the target operating characteristics, we optimized the initial eFuse configuration obtained by a VAE-based dual network. Genetic algorithm (GA) [4] is effective in the optimization especially because the relationship between input (eFuse configuration) and output (operating characteristics) is complex.

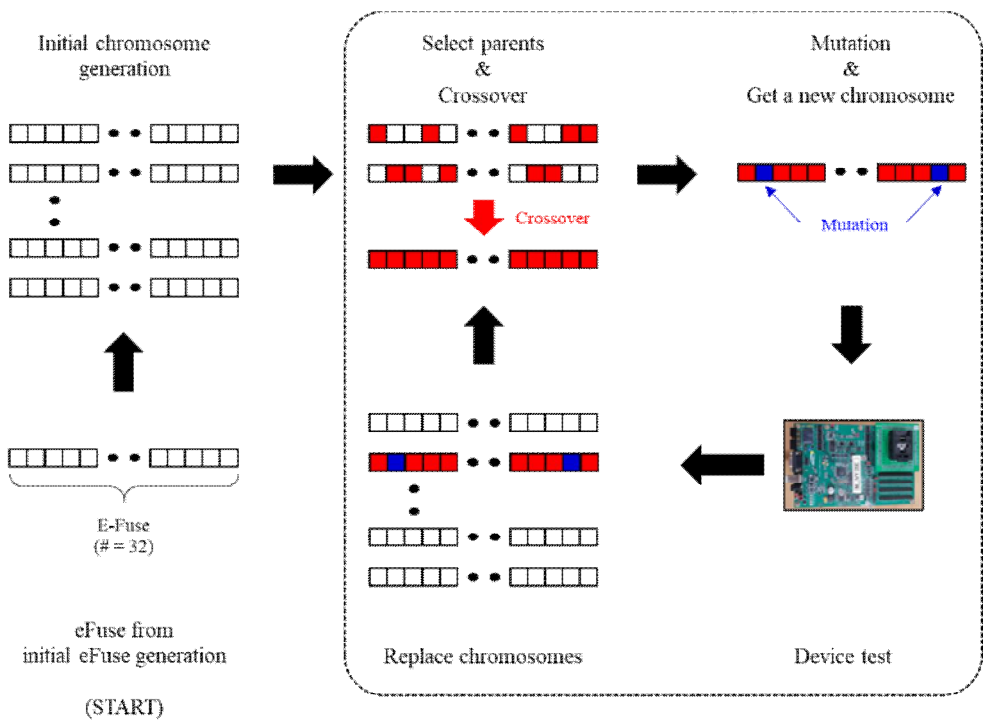


Figure 6. Test flow of eFuse optimization (GA)

Figure 6 shows the GA-based eFuse optimization. Given an initial eFuse configuration (obtained by the VAE-based dual network), we obtain an initial population of chromosomes, i.e., eFuse configurations and evaluate them. Then, we select chromosomes and perform cross-over and mutation to generate new chromosomes. We compare all the existing and new chromosomes and discard those having the worst fitness scores, which constitutes a generation. We try multiple generations to obtain the final results. We describe the details of each step below:

A *chromosome* corresponds to an eFuse configuration, i.e., a set of eFuses obtained by the VAE-based solution. In our experiments, the VAE-based dual network generates an eFuse configuration consisting of 32 8-bit eFuses. Thus, a chromosome consists of 32 8-bit values each (8-bit value) of which corresponds to a gene.

In order to generate the initial population, we modify the genes of the initial chromosome, i.e., the initial eFuse configuration, by perturbations, i.e., adding small random values (after selecting a random integer value in [-5, 5] in our experiments). We obtain N (= 50 in our experiments) chromosomes by perturbations.

We evaluate N chromosomes of the initial population, i.e., obtain N sets of operating characteristics, by configuring and measuring the NAND flash chip for each chromosome. In our experiments, we measure 24 values of the operating characteristics for each chromosome, i.e., eFuse configuration.

We obtain the *fitness score* of each chromosome as follows. First, we normalize each of the 24 operating characteristics across N configurations, e.g., normalizing the program time of the first word line (WL1) in N

configurations. Then, for each chromosome, we find the summation of its 24 normalized values to obtain the fitness score. Considering the score represents the operating characteristics of program time, the width of V_{th} cell distribution, and the upper bound of erase state, the lower the fitness score, the better chromosome.

We generate new chromosomes by selecting existing chromosomes and applying cross-over and mutation to them. We first select two chromosomes in a random manner. We denote a chromosome having the better (worse) fitness C_B(C_W). Then, we perform uniform cross-over [4] between them as follows. For each of gene positions (32 genes), we generate a random value in [0, 1]. If the random value is larger than a threshold (= 0.3 in our experiments), we select the corresponding gene of C_B and copy it to the corresponding position of a new chromosome. Otherwise, we copy the corresponding gene of C_W to the corresponding position of the new chromosome.

We apply mutation to the new chromosome we obtained from the cross-over. For each gene position, we generate a random number. If it is smaller than a threshold (=0.1 in our experiments), we add a random integer number in [-2,2] to the corresponding gene. Note that the gene has 8-bit integer. Thus, such a small random number slightly modifies the corresponding eFuse value.

We obtain a new generation by the above process of generating *M* new chromosomes and discarding the worst *M* ones. Note that we empirically obtained the parameters of the initial and additional population sizes (*N*=50 and *M*=10), cross-over (0.3), mutation (0.1 and [-2, 2]), and the number of

generations (10) via extensive experiments.

We generate M (= 10 in our experiments) new chromosomes by applying cross-over and mutation as explained above. Then, we obtain the fitness score of the new M chromosomes by configuring and measuring the NAND flash memory chip. Including the current population of N chromosomes, we compare total $N+M$ chromosomes and discard the M chromosomes having the worst fitness score thereby obtaining a new population of N chromosomes.

VI. EXPERIMENTAL RESULTS

6.1. Experimental Setup

We used an in-house test board equipped with a Samsung 64-stacked VNAND 512 Gb TLC product [12] that is actually in mass production. We selected 24 operating characteristics: PSUM in 10-word lines (WLs) including edge WLs + Disturb in 10 WLs including edge WLs + tPROG in 4 WLs. We also selected 32 8-bit eFuses related to the operating characteristics. Thus, a set of 24 (normalized) operating characteristics and 32 eFuses constitutes one training data. We obtained each training data by configuring the NAND flash chip with the eFuse configuration and measuring the 24 operating characteristics, which takes approximately 15 seconds on the board[3]. The size of the training set is 50,000 and that of the test set is 500. We designed on TensorFlow 3.0 [13] the VAE-based dual network consisting of a VAE encoder (24 inputs and 10 outputs), decoder (10 inputs and 24 outputs), and MLP (10-256-512-1024-2048-32 nodes for layers). We also trained an MLP (24-2048-1024-512-256-32) as the baseline, which accepts as input the operating characteristics and as the output generates eFuse configuration.

The target specification is pre-determined by experts. It is determined

---

3) Our board is not for a production test purpose. The advanced production test equipment will reduce by 5 times the time of configuration and measurement.

Table II. NAND flash characteristics variation

Items	Standard Deviation
PSUM	19 mV
Disturb	19 mV
tPROG	8 us

via estimation models during design time and via sampling a large number of flash chips at an early stage of production. Table II shows the target specification of PSUM, Disturb, and tPROG in terms of difference from the associated target mean values. In our experiments, we determined the target specification within 2.5 sigma of the distribution. Thus, a successful eFuse configuration gives PSUM, Disturb and tPROG within 50 mV, 50 mV, and 20 us of the associated mean, respectively.

6.2. Initial eFuse Generation Results

Figure 7 shows the test flow of initial eFuse generation. As explained in Section IV, in order to generate an initial eFuse configuration, the target operating characteristics is provided as input to the encoder that generates the latent vector Z. The MLP accepts as input the latent vector to generate the output of the initial eFuse configuration. Finally, we evaluate the initial eFuse configuration using the test board and obtain the actual operating characteristics.

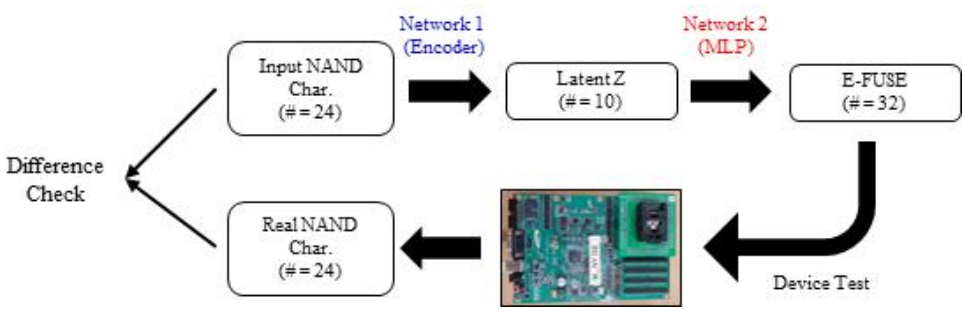
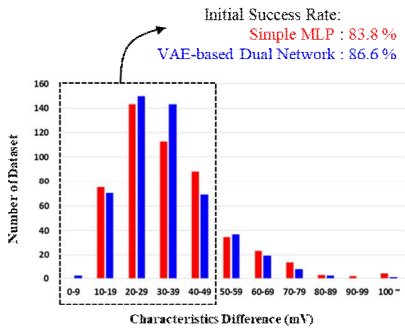
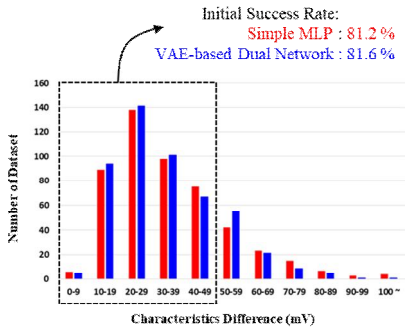


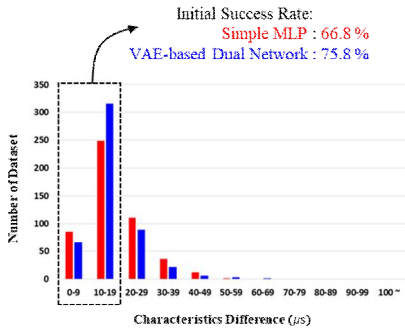
Figure 7. Test flow of initial eFuse generation



(a) Histogram of PSUM



(b) Histogram of Disturb



(a) Histogram of tPROG

Figure 8. Results of initial eFuse generation

Figure 8 shows the result of initial eFuse generation using the VAE-based dual network. We compare the difference (X-axis in the figure) between the target mean and the actual operating characteristics for PSUM (in mV), Disturb (in mV) and tPROG (in μs). The figure shows the histogram of 500 test data for each of the operating characteristics. The figure shows that the VAE-based dual network gives high-quality initial eFuse configurations where 86.6% (PSUM), 81.6% (Disturb), and 75.8% (tPROG), which is called *initial success rate*, of eFuse configurations satisfy the target specification denoted with the dashed boxes.

The figure also shows that the proposed VAE-based dual network outperforms the baseline MLP by providing by 2.8%, 0.4%, and 9.0% better initial success rate for PSUM, Disturb, and tPROG, respectively. Especially when the training dataset gets smaller, the proposed VAE-based dual network is better than the baseline MLP. Table III shows the drop of initial success rate when the size of the training set gets reduced from 50,000 to 30,000. As the table shows, the initial success rate of baseline MLP drops by 7.8%, 4.4%, and 9.4% while our proposed network gives much smaller drops by 4.6%, 1.2%, and 5.4%.

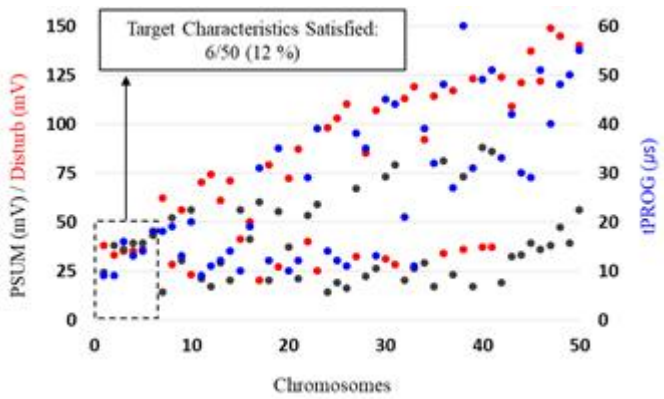
Table III. Accuracy drop according to training dataset

Item	VAE(50,000)		VAE(30,000)	MLP(50,000)		MLP(30,000)
PSUM	86.6%	-4.6% →	82.0%	83.8%	-7.8% →	76.0%
Disturb	81.6%	-1.2% →	80.4%	81.2%	-4.4% →	76.8%
tPROG	75.8%	-5.4% →	70.4%	66.8%	-9.4% →	57.4%

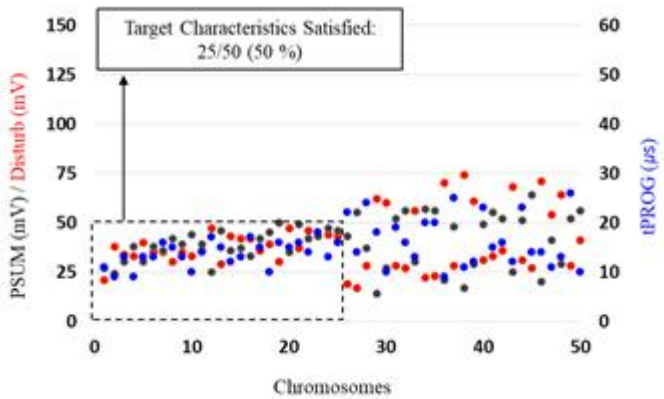
6.3. eFuse Optimization Results

We evaluated the GA-based eFuse optimization by choosing the outlier, i.e., worst-case eFuse configuration obtained by the VAE-based dual network. In Section V, given an initial eFuse configuration, the GA solution first applies perturbation to generate 50 eFuse configurations, i.e., chromosomes to form the first generation. Then, for each new generation, it generates 10 new chromosomes, evaluates the total population of 60 chromosomes and discards 10 chromosomes having the worst fitness score. We processed 10 generations with this technique. We evaluated total 150 (50 initial and 100 new) chromosomes, i.e., eFuse configurations on the test board equipped with the VNAND flash chip.

Figure 9(a) shows the results of the NAND flash chip test board evaluation of 50 initial eFuse configurations. The Y-axis represents the difference between the target and the actual operating characteristics and the X-axis represents the sorted index of the corresponding chromosome. The dashed box represents the region of within specification. The figure shows that 12% of eFuse configurations obtained by the perturbation already meet the specification, which also shows that the operating characteristics of the initial eFuse configurations are not far from the target specification. Figure 9(b) shows the results of 50 eFuse configurations obtained by the GA solution. The figure shows that the GA solution proves effective and increases from 12% to 50%, the success rate of eFuse configurations, which shows that the GA solution has the potential of optimizing even outlier initial eFuse configuration.



(a) The result: Initial chromosome generation



(b) The result: after genetic algorithm

Figure 9. Results of eFuse optimization (GA)

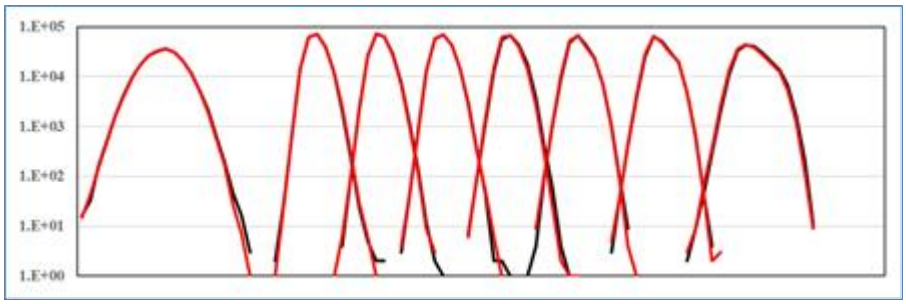


Figure 10. Cell V_{th} distribution result

The quality of the eFuse configuration can be evaluated by measuring V_{th} cell distribution. Figure 10 compares the V_{th} cell distribution of one of the final eFuse configurations and that of the target specification. The figure shows that the V_{th} cell distribution (in red) of the obtained eFuse configuration is almost indistinguishable from the target V_{th} cell distribution (in black). The figure proves the quality of the eFuse configuration generated by the proposed method.

Table IV. Time consuming of experiments

Items	Time Consuming
Dataset Generation (# = 50,000)	About 9 Days ⬇
Training VAE-based Dual Network	1.0 hour ⬇
Test and Check the Result of Initial eFuse Generation	3 mimute ⬇
Proceeding with eFuse Optimization	1.0 hour ⬇

6.4. Discussion

Our proposed method can significantly reduce the design time of the eFuse configuration. Typically, in the current design practice, the discovery of eFuse configurations for the given target specification occurs within some weeks. Our experiments, which includes training data acquisition, and eFuse generation/optimization were automated, and took within nine days, most of which was spent to configure and measure the flash memory chip (Table IV). In the mass production setting, the advanced test equipment can offer a 5 times faster configuration[4] and measurements, than our test board thereby making eFuse configuration a two-day job in mass production[5].

---

4) The target board used in our experiments has limited memory bandwidth, which makes configuration and measurement take 15 seconds. The production test equipment can be used to resolve this problem to enable much faster (> 5) configuration and measurement.

In our experiments, we report the design time of eFuse configuration for a single NAND flash chip. The real eFuse configuration for mass production is performed by sampling flash memory chips and (the experts) finding eFuse configurations for the sampled chips. In such a real setting, our proposed method can be utilized to reduce the eFuse configuration time for each of sampled flash memory chips.

Considering that the target operating characteristics vary across different products, e.g., server, PC, automotive, mobile, etc., the same NAND flash chip can be configured for different operating characteristics, which complicates eFuse configuration owing to the immense design space of the operating characteristics for exploration. Our proposed method has the potential of screening the validity of the explored operating characteristics thereby reducing the search space. It is because the VAE network has the inherent characteristics that it can produce high fidelity result between the input and output for an input similar to the training dataset, while it cannot guarantee such a fidelity if the input is significantly far from the distribution of training dataset. Thus, in the case of an excessive gap between input and output, it is probable that the desired operation characteristics (used as the input) are not realizable with the flash memory chip. In our future work, we will investigate the feasibility of this idea to facilitate the design space exploration of the operating characteristics.

---

5) We sample chips after finishing the eFuse trimming which determines analog bias level, inter-chip variation compensation, etc.

VII. RELATED WORK

This section reviews eFuse trimming methods for the analog circuit used in semiconductor chips and recent works on machine learning application to chip test. To the best of the authors' knowledge, there is no previous work on automatic eFuse optimization.

In [14], Bongale et al. proposed an eFuse trimming technique to compensate for the post-fabrication variations of analog circuits. It is a formula-based search method to overcome the problem of iterative evaluations due to the interdependent characteristics between trim codes. Another study [15] focused on the fact that each chip on the wafer is assigned a different trim code for post fabrication calibration. To reduce the test process time, the authors proposed to predict the initial trim value of each chip. In both [14] and [15], the authors discussed techniques for compensating for inter-chip process variation to meet the target operating characteristics of individual analog circuits determined during design time. Our work is different from those studies [14] [15] because we address the hard eFuse configuration which is characterized by complex inter-dependency among the operating configurations thereby simultaneously involving multiple circuits while the existing works target individual circuits.

Recently, there are studies in test engineering to increase the defect screen accuracy using the autoencoder (AE) method. Lin and Cheng introduced an AE-based method of test escape screening [16]. AE uses an encoder to compress the input data and recovers the input data through the decoder again. Assuming that the characteristics of the input differ from that

of training data, the output tends to fail to recover the input. By applying this feature of AE, the authors propose training the AE network with only good chip characteristics, and aim at identifying a defect chip by using the fact that the AE network can fail to recover its input in case of a defect chip. In a study [17], the AE-based fail detection method used in research [16] is refined to adopt the VAE method in order to prevent overfitting due to the lack of training data. In addition, the authors propose an anomaly score method based on probability distribution which sets the criteria for the identification of fail chips.

Genetic algorithm (GA) has been used to improve the test coverage. In [18] and [19], the authors presented GA-based methods of generating the test samples. In our proposed method, GA plays the role of refining the initial eFuse configuration to maximize the success rate of eFuse configuration.

VIII. CONCLUSION

In this paper, we proposed a novel methodology of automating eFuse configuration in NAND flash chips. It consists of two steps, the VAE-based dual network for initial eFuse configuration and GA-based optimization. Given the target operating characteristics, our VAE-based dual network provides an initial eFuse configuration. The GA-based solution refines the initial eFuse configuration to meet the target operating characteristics. We evaluated the proposed method using a 512 Gb 64-stacked V-NAND chip. Our experiments show that the proposed two-step method has the potential of a significant reduction in design time from weeks of eFuse configuration to a two-day job in a mass production setting. The experiments also show that, compared to the MLP baseline, the VAE-based network offers a better quality of eFuse generation with the same amount of training data and the GA-based solution proves that even an outlier initial eFuse configuration can be optimized to successful ones at a small cost.

REFERENCE

[1] S. K. Park, “Technology scaling challenge and future prospects of DRAM and NAND flash memory,” IEEE International Memory Workshop (IMW), 2015, pp. 1-4.

[2] N. Robson, J. Safran, C. Kothandaraman, A. Cestero, X. Chen, R. Rajeevakumar, and et al., “Electrically Programmable Fuse (eFuse): From Memory Redundancy to Autonomic Chips,” IEEE Custom Integrated Circuits Conference, 2007, pp. 799-804.

[3] D. P. Kingma and M. Welling, “Auto-encoding Variational Bayes,” International Conference on Learning Representations (ICLR), 2014.

[4] D. Whitley, “A genetic algorithm tutorial,” Statistics and Computing, 1994, vol. 4, pp. 65-85.

[5] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, “Flash Memories,” MA, Boston:Kluver, 2000.

[6] A. Torsi, Y. Zhao, H. Liu, T. Tanzawa, A. Goda, P. Kalavade, and et al., "A Program Disturb Model and Channel Leakage Current Study for Sub-20 nm NAND Flash Cells," IEEE Transactions on Electron Devices, Jan 2011, vol. 58, no. 1, pp. 11-16.

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, 2015, vol. 521, pp. 436-444.

[8] S. Lawrence, C. L. Giles, and A. C. Tsoi, “Lessons in neural network training: Overfitting may be harder than expected,” Fourteenth National Conference on Artificial Intelligence (AAAI), 1997, pp. 540–545.

[9] M. Hoffman, D. Blei, J. Paisley, and C. Wang, “Stochastic variational inference,” The Journal of Machine Learning Research, 2013, vol. 14, pp. 1303–1347.

[10] S. Kullback, “Information theory and statistics,” Dover Publications, 1968.

[11] T. M. Cover and J. A. Thomas, “Elements of information theory,” New York:Wiley, 1991.

[12] C. B. Kim, D. H. Kim, W. P. Jeong, H. J. Kim, I. H. Park, H. W. Park, and et al., “A 512-Gb 3-b/Cell 64-Stacked WL 3-D NAND Flash Memory,” IEEE Journal of Solid-State Circuits, 2018, vol. 53, no.1, pp. 124-133.

[13] M. Abad, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean and et al., “Tensorflow: A system for large-scale machine learning,” 2016.

[14] P. Bongale, V. Sundaresan, P. Ghosh, and R. Parekhji, "A novel technique for interdependent trim code optimization," IEEE 34th VLSI Test Symposium (VTS), 2016, pp. 1-6.

[15] C. Xanthopoulos, A. Ahmadi, S. Boddikurapati, A. Nahar, B. Orr, and Y. Makris, "Wafer-level adaptive trim seed forecasting based on E-tests," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4.

[16] F. Lin and K.-T. Cheng, “An artificial neural network approach for screening test escapes,” IEEE/ACM Asia and South Pacific Design Automation Conference, 2017, pp. 414–419.

[17] M. Shintani, M. Inoue, and Y. Nakamura, “Artificial Neural Network Based Test Escape Screening Using Generative Model” IEEE International Test Conference (ITC), 2018, pp. 1-8.

[18] R. Pargas, M. J. Harrold, and R. Peck, “Test-Data Generation Using Genetic Algorithms,” Software Testing, Verification and Reliability, 1999, pp. 263-282.

[19] T. Golonek and J. Rutkowski, “Genetic-Algorithm-Based Method for Optimal Analog Test Points Selection,” IEEE Transactions on Circuits and Systems II: Express Briefs, Feb. 2007, vol. 54, no. 2, pp. 117-121.

국문 초록

메모리 공정 기술이 발전하고 비즈니스 시장이 다양해 짐에 따라 웨이퍼 수율을 높이고 비즈니스 특성 목표를 만족하기 위한 후 공정 과정이 매우 중요해 지고 있다. 전기적 퓨즈 조절 방식(이-퓨즈 최적화 및 트림)은 메모리 칩 후 공정 과정에서 사용되는 전통적인 방식이다. 엔지니어는 이-퓨즈 조절을 통해 웨이퍼 상의 칩들 간의 초기 특성의 변화를 보상하거나, 신뢰성, 레이턴시, 파워 소모, 그리고 I/O 대역폭 등의 칩 목표 특성을 보장한다. 이-퓨즈 조절 업무는 다수의 숙련된 엔지니어가 필요하고 또한 상당히 많은 시간을 소모한다. 본 논문에서는 낸드 플래시 칩의 동작 특성 목표를 얻기 위한 기계 학습 기반의 이-퓨즈 자동 생성 기술을 제안하고, 해당 기술은 엔지니어의 작업시간을 획기적으로 단축시킬 수 있다. 논문의 기술은 두 단계로 구성 된다. 첫 번째 단계에서는 variational autoencoder (VAE) 기술을 적용하여 목표하는 동작 특성을 만족시키는 초기 이-퓨즈 구성을 생성한다. 두 번째 단계에서는 유전 알고리즘을 적용하여 초기 생성된 이-퓨즈 구성에 대하여 목표하는 성능 특성과의 정합성을 추가로 개선하여 최종적으로 목표하는 성능 특성을 얻는다. 논문의 평가는 실제 양산중인 삼성 64단 브이낸드 제품을 이용하여 진행하였다. 논문의 이-퓨즈 자동화 생성 기술은 2일 이내의 구현 시간만이 소요된다.

주요어 : 낸드 플래시, 이-퓨즈, 최적화, 기계 학습, 딥 러닝, VAE, 유전 알고리즘

학번 : 2017-29459